Sabanci University

Faculty of engineering and Natural Sciences

CS 300 Data Structures

Assigned: **Nov 19, 2020** Due: **Nov 30, 2020 @ 11:55pm**

PLEASE NOTE

SOLUTIONS HAVE TO BE YOUR OWN. NO COLLABORATION OR COOPERATION AMONG STUDENTS IS PERMITTED.

10% PENALTY WILL BE INCURRED FOR EACH DAY OF OVERTIME. SUBMISSIONS THAT ARE LATE MORE THAN 3 DAYS WILL NOT GET ANY CREDITS.

SUBMISSIONS WILL BE MADE TO THE SUCOURSE SERVER. NO OTHER METHOD OF SUBMISSION WILL BE ACCEPTED.

Introduction

Hanan Samet defines point quadtrees as “a multidimensional generalization of a binary search tree” to store and query 2D points. For further details, see the attached document taken from his book titled “Design & Analysis of Spatial Data Structures.”

In this assignment, you will implement a point quadtree of cities. More specifically, given a 2D space and a list of cities, each of which is represented as a 2D point, you will insert the cities in the order they are given into an initially empty point quadtree. Then, given a query point *A* in the same space together with radius *r,* you will find all the cities within radius *r* of *A*. For further details about point quadtrees and the way the insert and find operations should be implemented, see the attached document.

Note that we will use this assignment to evaluate whether you can use the knowledge you have gained in the lectures to understand and implement related data structures. Therefore, we opt not to share any further information, but a textbook definition of the point quadtrees.

Your program should read two text files, namely **cities.txt** and **queries.txt**. While the former specifies a two dimensional space together with some cities located in this space, the latter specifies the queries to be answered.

In the cities.txt file, the first line specifies the 2D space by presenting the coordinate of the upper left corner of the space in the form of

```
<x coordinate> <y coordinate>
```

Note that the coordinate of the origin, which, in this particular context, corresponds to the lower left corner of the space, is `(0, 0)` and that negative coordinate values are not valid.

Each of the remaining line in the cities.txt file then represents a city in the form of:

```
<city_name> <x coordinate of the city> <y coordinate of the city>
```

City names do not contain space characters and the coordinates cannot assume negative values.

Given this file, your program shall insert the cities into an initially empty point quadtree in the order they are given and then shall pretty print the tree using the following recursive algorithm

```
pretty_print (root):   // pretty print the quadtree rooted at root:
    if root != NULL:   // if the tree is not empty
        print the city name stored at the root
        pretty_print(root.SE) // recursively print the south east subtree
        pretty_print(root.SW) // recursively print the south west subtree
        pretty_print(root.NE) // recursively print the north east subtree
        pretty_print(root.NW) // recursively print the north west subtree
```

Once the point quadtree is constructed, the queries.txt file is read. In this file, every lines corresponds to a query in the form

```
<x coordinate> <y coordinate> <radius>
```

Given such a query, two lines should be produced.

The first line shall report the comma separated list of cities within `<radius>` unit of distance from the point `(<x coordinate>, <y coordinate>)`. And , the second line shall report the comma separated list of cities visited during the search operation. Note that in each case cities should be reported in the order they have been found and visited, respectively; you shall visit the quadrants (i.e., subtrees) **recursively** in exactly the same order given in Figure 2.17 of the attached document. Note further that the result of a query should be printed out right after the query line is processed. Furthermore, if the no cities are found, then the string '<None>' should be printed out.

Sample Run

Given the following cities.txt file:

```
100 100
Chicago 35 42
Mobile 52 10
Toronto 62 77
Buffalo 82 65
Denver 65 45
Omaha 27 35
Atlanta 85 15
Miami 90 5
```

The following quatree shall be constructed (as illustrated in Figure 2.4 of the attached document):

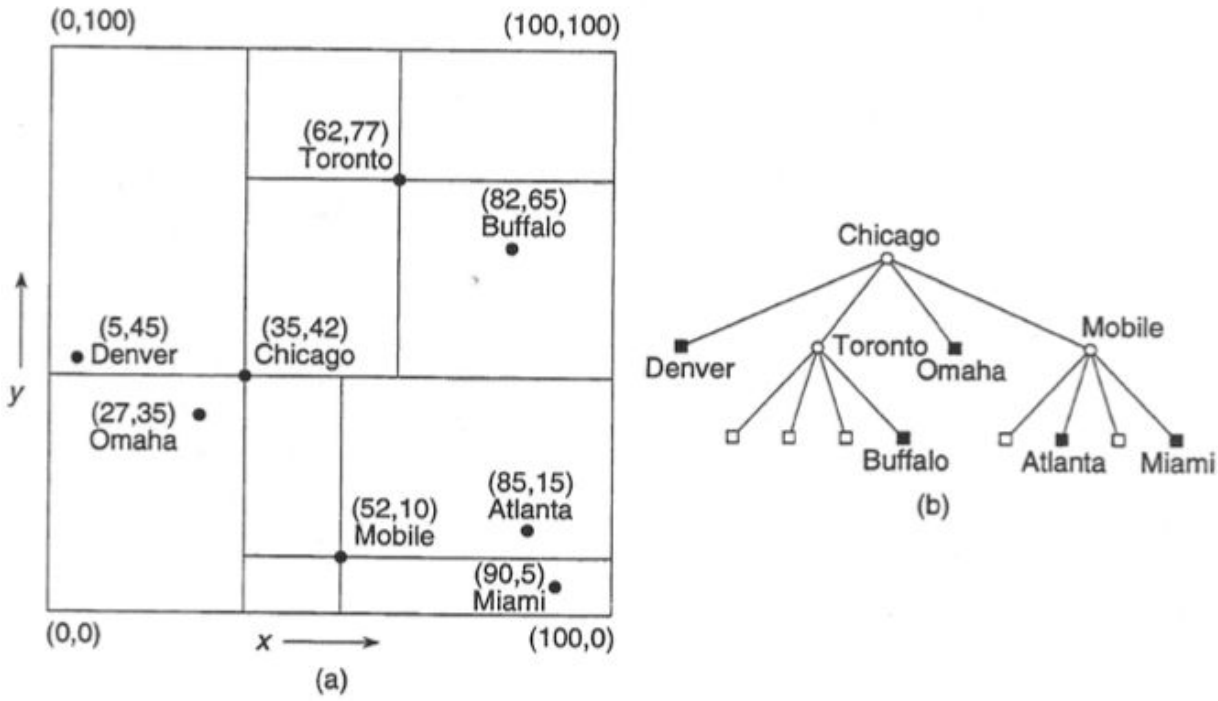

Figure 1 (a) The resulting partition of space and (b) the tree representation.

Thus, the following output should be printed out:

```
Chicago
Mobile
Miami
Atlanta
Omaha
Toronto
Buffalo
Denver
```

Then given the following query.txt file:

```
83, 10, 8
25, 33, 10
42, 83, 2
82, 35, 32
62, 77, 24
```

The output should be as follows:

```
Atlanta
Chicago, Mobile, Miami, Atlanta

Omaha
Chicago, Mobile, Omaha, Denver

<None>
Chicago, Toronto

Miami, Atlanta, Buffalo
Chicago, Mobile, Miami, Atlanta, Toronto, Buffalo

Toronto, Buffalo
Chicago, Toronto, Buffalo
```

Submission

Your code should be submitted to SUCourse+ at the deadline given on the first page. You should follow the following steps:

- Name the folder containing your source files as *XXXX-NameLastname* where *XXXX* is your student number. Make sure you do NOT use any Turkish characters in the folder name. You should remove any folders containing executables (Debug or Release), since they take up too much space.
- Compress your folder to a compressed file named, for example, *5432-AliMehmetoglu.zip*. After you compress, please make sure it uncompresses properly and reproduces your folder exactly.
- You then submit this compressed file in accordance with the deadlines above. Your homework will be graded in the following way:
    - If your program does not construct and query quadtrees as described in the attached document, you will get 0 points. This will be the case even if your program works correctly otherwise.
    - We will run about 10 tests on your homework. Each correct test will earn you 10 points. Note that your outputs should be in the exact same format we described above.

Good luck